منتديات جامع العوم .

دورة تصميم برنامج تسيير مخزن .

مخطط الفئات .

إعداد شنيقي نصرالدين

سنتطرق في موضوعنا هذا إلى مخطط الفئات بعد انتهائنا من مخطط حالات الاستخدام ، لكن قبل الخوض في مناقشة هذا المخطط علينا أولا التطرق إلى بعض المفاهيم المتعلقة بهذا المخطط و بالتحديد مفهوم البرمجة الكائنية المنحى (POO: Programmation orientée objet) لأنها الأساس الذي يقوم عليه مخطط الفئات ، لنبدأ على بركة الله .

1. البرمجة كائنية المنحى :

تمهيد : يحاول الإنسان دوما تغيير أسلوب حياته و تحديثه و تطويره باستمرار و ذلك طلبا للرفاهية و العيش الحسن و اختصارا للوقت الثمين الذي يعد بالنسبة للإنسان أهم شيء في حياته و البرمجة هي واحدة من المجالات التي طالها هذا التطوير و التحسين حيث مرت بثلاث مراحل مهمة أثناء تطورها :

- مرحلة البرمجة بلغة الآلة: في هذه المرحلة كان المبرمجون يكتبون برامجهم باستخدام رقمين فقط هما الصفر و الواحد و كانت البرامج التي تكتب بهذه اللغة جد معقدة و من الصعب فهمها.

- مرحلة البرمجة بلغة التجميع langage d'assemblage : أتت هذه اللغة لتزيل بعض تعقيدات لغة الآلة فهي ليست إلا اختصارا لتعليمات ست عشرية هي في الأصل صفر وواحد، ساهمت هذه اللغة في التطور السريع للغات البرمجة إلا أنها مازالت تحتوي على بعض التعقيدات و التعليمات التي توحي للمبرمج بأنه مازال يتعامل مباشرة مع المكونات الداخلية للحاسب .

- مرحلة البرمجة الإجرائية La programmation Procédurale: مع ظهور هذا الجيل من لغات البرمجة بزغ فجر اللغات عالية المستوى و التي تشبه تعليماتها لغة الإنسان فهي تحتوي كلمات باللغة الإنجليزية كـ BEGIN و END و While و IF و غيرها.و بظهور هذه اللغات أصبحت البرمجة في متناول الجميع حتى الهواة من المبرمجين، في الوقت الذي كانت فيه لغات البرمجة منخفضة المستوى(لغة الآلة و لغة التجميع) مقتصرة على دوي الاختصاص. لكن هذا الجيل من لغات البرمجة كان يعتمد على الإجراءات و الدوال التي كانت تطبق عددا من التعليمات على مجموعة من المعطيات أو البيانات ، و كانت البرامج التي تكتب بإحدى لغات البرمجة الإجرائية غير منظمة وتفتقر للتنسيق بل كانت عبارة عن عدد كبير من الدوال و الإجراءات المتداخلة التي يصعب أحيانا تذكرها و العمل بها . و أبسط مثال عن المكتبات البرمجية التي تعتمد على أسلوب البرمجة هذا هي مكتبة ال Win 32 API التي تُستعمل للتحكم بالويندوز مكتوبة بلغة السي و هي تتكون من عشرات الآلاف من الدوال التي يستحيل تذكرها .

- مرحلة البرمجة الكائنية : يعد مفهوم البرمجة كائنية المنحى أساس لغات البرمجة في وقتنا الحالي فجميع اللغات الحديثة تعتمد عليه ، حيث أن لغة السي ++ و الجافا و السي شارب و غيرها كلها لغات كائنية المنحى.

مفهوم البرمجة كائنية المنحى جاء بعد البرمجة الإجرائية التي كانت تفتقر للتنظيم و التنسيق ، من هنا انطلقت البرمجة كائنية المنحى فقد جاءت لكي تجمع بين المعالجة التي تقوم بها الإجراءات و الدوال و بين البيانات و المعطيات التي تحتاج لهذه المعالجة ، فبعد أن كانت هذه الإجراءات مفصلة تماما عن البيانات (يمكن أن تقوم بتعريف بيانات برنامجك في أوله و الإجراءات في وسطه أو آخره)، أصبحت الآن في البرمجة كائنية المنحى مدمجة مع بعضها في نفس المكان الذي يدعى الكائن Objet (ربما يبدو لك هذا المصطلح جديدا و غريبا ، لكنه في الحقيقة أساس البرمجة كائنية المنحى ، ذلك أن مخترعي هذه البرمجة ركزوا على محاكاة الواقع للاستفادة من كونه الشيء الذي يهتك به الإنسان كثيرا) ، فأصبح لكل كائن من الكائنات التي تشكل في مجموعها برنامجك بياناته (خصائصه) و دواله أو إجراءاته الخاصة به . فلا يمكن لدالة أو إجراء ينتمي لكائن آخر بالتدخل و التعديل على بيانات هذا الكائن مباشرة ، إذا فأول مفهوم من مفاهيم البرمجة كائنية المنحى هي الكائنات :

الكائن (Objet) : في البرمجة كائنية المنحى يمكن اعتبار البرنامج عبارة عن مجموعة من الكائنات التي تتفاعل فيما بينها من أجل القيام بوظائف هذا البرنامج ، كل كائن من هذه الكائنات يحتوي على خصائص (Propriétés) سوءا كانت هذه الخصائص عبارة عن صفات مميزة (Attributs) للكائن أو عبارة عن سلوكيات (comportement) يتصف به .

كما قلت آنفا الكائن يتميز بعدة خصائص هي :

- المُعرف أو الهوية(Identité): كل كائن يحتوي على معرف يسمح له بالتمايز عن غيره من الكائنات، فمثلا يمكن أن نميز المنتج Produit و هو كائن عن باقي المنتجات بالرمز Code و كائن السيارة برقم السيارة التسلسلي Numéro d'immatriculation.

- **الصفات (Attributs) :** هي عبارة عن معلومات تميز الكائن ، و بعبارة برمجية هي عبارة عن متغيرات تخزن معلومات حول حالة الكائن .

- **الطرائق (Les méthodes) أو العمليات Opérations :** في الحقيقة في البرمجة كائنية المنحى نستعمل لفظ méthode أو opération عوضا عن fonction أو procédure ذلك أنها أصبحت كما قلنا مدمجة في الكائن.

الطرائق هي السلوكيات التي يتصف بها الكائن أو مجموعة العمليات القادر على انجازها و هي تسمح للكائن بالتفاعل مع باقي الكائنات ، كما أنها مرتبطة بقوة بالصفات Attributs لأنها غالبا ما تقوم بالتعديل عليها.

الفئة La classe : الفئة هي نوع مجرد من بيانات Type de données abstrait لديها خصائص propriétés مشتركة بين جميع أفراد عائلة من الكائنات و بإمكانها إنشاء كائنات تمتلك هذه الخصائص.يمكن القول أن الفئة هي النموذج العام لنوع من الكائنات فمثلا فئة الطلبة تحتوي على صفة: الاسم ، اللقب ، رقم الطالب ، سنة الطالب ... من هذه الفئة يمكن أن ننشأ نسخة لطالب اسمه نصرالدين ، لقبه شنيقي ، رقمه :1546552 ، سنة دراسته أولى ماستر ..

الكبسلة Encapsulation : أو ما يعرف أيضا بالتغليف و هي تنص على إخفاء تفاصيل عملية من العمليات التي يقوم بها الكائن و إبراز ما يعرف بالواجهة فقط (Interface) و ذلك ليتسنى لمستخدم الفئة معرفة المهام التي يمكن لفئته القيام بها من دون الخوص في تفاصيلها . فمثلا إذا كانت لديك فئة تحتوي على الدوال الرياضية كالجب و جيب التمام و غيرها ، أنت كمستعمل لهذه الفئة لن تضطر لمعرفة كيف يُحسب الجيب مثلا ، بل يكفي أن تمرر القيمة التي تريد الاستعلام عنها فقط إلى عملية الجيب و التي بدورها تتولى عوضا عنك حساب هذه القيمة و تقدمها إليك جاهزة للاستعمال. و هنا يظهر مفهوم مبرمج الفئة و مستعمل الفئة فمبرمج الفئة سيهتم بجميع تفاصيل الفئة و يقوم ببرمجة جميع عملياتها. أما مستعمل الفئة ما عليه سوى استعمال هذه العمليات الجاهزة.

هناك فائدة أخرى للكبسلة و هي حفظ بيانات الكائن من التغيير من طرف كائن آخر مباشرة و ذلك لتجنب تداخل و تعارض البيانات. و بهذا يصبح الكائن مستقلا تماما عن بقية الكائنات و يصبح له كيان خاص ، فلو أردنا إجراء تغييرات على كائن من الكائنات التي تشكل برنامجنا فنحن سنهتم بهذه التغييرات على مستوى هذا الكائن فقط من دون الخوف من تأثير هذه التغييرات ، و هذا ما لم يكن في البرمجة الإجرائية التي يؤدي فيها تغيير بسيط في متغير من متغيرات البرنامج أو في دالة من دواله إلى التأثير على عدد كبير من الدوال و المتغيرات التي كانت في حالة ترابط وثيق فيما بينها .

الوراثة Héritage : الوراثة في البرمجة تشبه إلى حد كبير الوراثة في الواقع ، فنقول عن فئة أنها ترث فئة أخرى هذا يعني أن جميع خصائص الفئة الموروثة ستنتقل إلى الفئة الوارثة (ما يقابلها في الطبيعة جميع ممتلكات الموروث تنتقل إلى الورثة).

كانت هذه بعض مفاهيم البرمجة الكائنية المنحى فيما يلي سنقوم بشرح مخطط الفئات و أهم خصائصه .

2. مخطط الفئات Diagramme de Classes :

يعتبر هذه المخطط أهم مخططات UML و هو المخطط الذي لا ينبغي أن يغيب عن أي نمذجة لأي نظام يعتمد على البرمجة كائنية المنحى.

و في حين أن مخطط حالات الاستخدام يوضح لنا النظام من وجهة نظر الممثلين Les acteurs، فإن مخطط الفئات يوضح لنا البنية الداخلية للنظام.هذا المخطط يوفر لنا تمثيلا مجردا لكائنات النظام التي تتفاعل فيما بينها من أجل تنفيذ مخططات حالات الاستخدام.

1.2. تمثيل الفئة في مخطط الفئات :

الفئة تمثل في هذا المخطط بالشكل التالي :

```
Nom_de_la_Class
+Attribute1
-Attribute2
+...
#AttributeN
-Operation1()
+Operation2()
+...()
+OperationN()
```

الشكل مقسم إلى ثلاث أجزاء ، في الجزء الأول نقوم بكتابة اسم الفئة و يكون اسم الفئة دال على الوظيفة التي تؤديها ، في الجزء الثاني نقوم بسرد جميع الصفات التي تتصف بها فئتنا و في الجزء الأخير نقوم بإدراج جميع الطرائق أو العمليات التي تقوم بها الفئة .

2.2. مجال الرؤية La visibilité : ربما قد لاحظت علامة + و – و # ، هذه العلامات تبين لنا إمكانية الوصول إلى الـ Attribut أو الـ méthode التي بداخل الفئة ، و هي تعرف بمحددات الوصول :

محدد الوصول Public (+) : أي عنصر من الفئة أو من خارج الفئة يمكنه رؤية الخاصية (الصفة Attributs أو العملية opération) التي تسُبق بهذا المحدد.

محدد الوصول Private (-): لا يمكن رؤية الخاصية التي تسُبق بهذا المحدد إلا من داخل الفئة نفسها.

محدد الوصول Protected (#): لا يمكن رؤية الخاصية التي تسبق بهذا المحدد إلا من داخل الفئة أو من فئة أخرى ورثت هذه الفئة.

مثال : لنأخذ على سبيل المثال فئة السيارة ، سيكون تمثيلها في مخطط الفئات كما يلي:

```
Voiture
+Marque
+Modèle
-DistanceEcoulée
+Conduire(int Km)
```

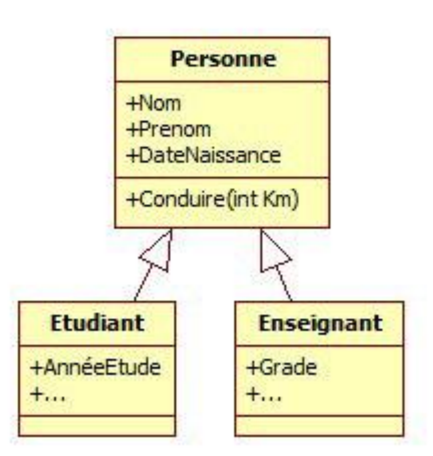
هذه الفئة لديها ثلاث صفات هي النوعية La marque، الطراز Le modèle ، المسافة التي قطعتها La distance écoulée، إضافة إلى العملية ()Conduire و التي تمثل سلوك القيادة ، و من المعلوم أن السيارة كلما قطعت مسافة معينة يقوم عداد خاص في السيارة بحساب هذه المسافة ، أي أن هذه العملية تقوم بالتأثير أو التعديل على الصفة DistanceEcoulée حيث و في كل مرة يتم استدعاء العملية Conduire نقوم بإضافة قيمة Km إلى قيمة DistanceEcoulée القديمة .

و ربما تلاحظ أن الصفة DistanceEcoulée مسبوقة بمحدد الوصول Private ذلك لنمنع أي تعديل لهذه الصفة من خارج الفئة Voiture ، و نحصر هذا التعديل فقط من خلال استدعاء العملية ()Conduire و هذا ما يمكن أن نترجمه في الواقع إلى أن المسافة التي تقطعها السيارة لا يمكن أن تتغير إلا إذا تحركت هذه السيارة فعلا .

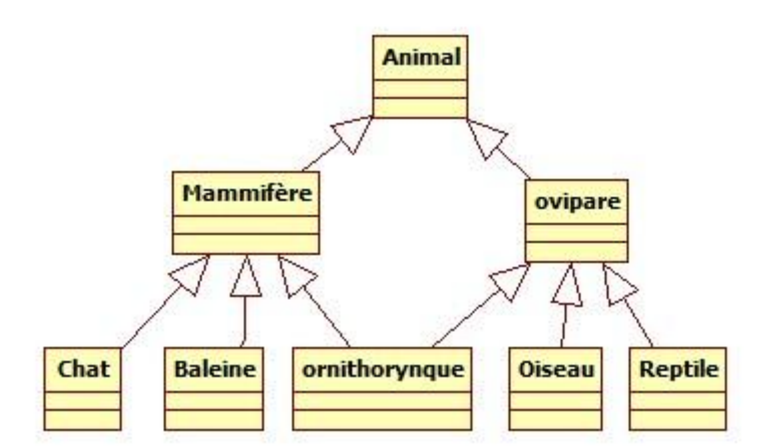
3.2. العلاقات بين الفئات :

1.3.2. التعميم و الوراثة Généralisation et Héritage :

سبق لنا و أن شرحنا مفهوم الوراثة في البرمجة كائنية المنحى . في مخطط الفئات نمثل علاقة الوراثة بين فئتين بسهم رأسه مثلث يشير إلى الفئة الموروثة و بدايته من الفئة الوارثة ، الفئة الوارثة تحتوي جميع خصائص الفئة الموروثة ماعدا الخصائص الخاصة أي التي تكون مسبوقة بمحدد الوصول Private . المثال التالي يبين لنا هذه العلاقة بين الفئة Personne و الفئتين Étudiant و Enseignant



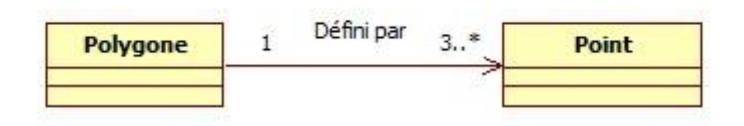
يمكننا القول أن الفئة Personne أعم من الفئتين Étudiant و Enseignant أو الفئتين السالفتين الذكر أخص من الفئة Personne أي كلما اتجهنا إلى الأعلى زاد التعميم و كلما اتجهنا إلى الأسفل زاد التخصيص و يظهر هذا جليا في حالة كون هناك أكثر من علاقة تعميم واحدة كالمثال التالي :



2.3.2. الروابط Les Associations:

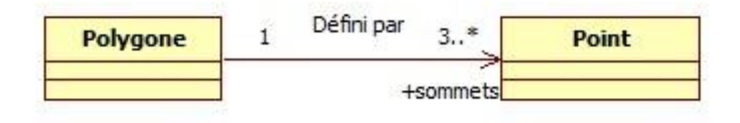
الروابط توضح لنا طريقة ارتباط الفئات فيما بينها و العلاقات التي تجمع بين مختلف الفئات.

مثلا في المثال التالي فئة المضلع Polygone معرفة على الأقل بثلاث نقاط أو أكثر أي بينها و بين الفئة Point رابطة Défini par (معرف بـ)



- نهاية الرابطة Terminaison d'association :

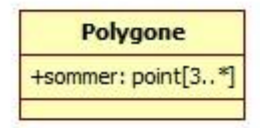
عرفنا فيما سبق أن الفئة تحتوي على صفات Attributs هذه الصفات يمكن أن تكون عبارة عن متغير من نوع سلسلة حروف string أو متغير صحيح int أو أي نوع من المتغيرات التي نعرفها في لغات البرمجة ، الجديد هنا هو أن الـ Attribut يمكن أن يكون نوعا جديدا من المتغيرات قمنا نحن بإنشائه كأن يكون نسخة من فئة عرّفناها سابقا ، و عليه يمكننا تمثيل هذا المتغير لما نستعمله في مخطط الفئات بوضع اسمه نهاية الرابطة بجانب الفئة التي استنسخناه منها ، و بالعودة للمثال السابق الفئة polygone تحتوي ثلاث نسخ أو أكثر من الفئة Point تشكل رؤوس Les sommets لهذا الـ polygone ، و يمكننا تمثيل هذه العلاقة في مخطط الفئات كما يلي :



و لكي تبسط الفكرة أكثر الكود التالي المكتوب بلغة السي شارب يوضح طريقة تحويل المخطط السابق إلى كود برمجي . سنقوم بالتطرق لهذا الموضوع في الأخير:

```
class Polygone
    {
        public Point[] Sommets = new Point[X];//3 حيث X من أكبر
    }
```

و التمثيل السابق مطابق للتمثيل التالي :



- التعدد La multiplicité :

ربما لاحظت في المثال السابق أن فئة polygone أمامها رقم 1 و فئة Point أمامها المجال 3..* في الحقيقة هذه هي الـ Multiplicité أي أنه أي نسخة من فئة الPolygone (أي كائن من الكائنات التي يمكننا إنشائها من هذه الفئة) يمكنه أن يحتوي ثلاث نسخ من الفئة Point فأكثر تشكل رؤوس هذا الـ polygone و أن أي نسخة من الفئة Point تنتمي إلى نسخة واحدة و وحيدة من الفئة Polygone.

و هذه بعض الأمثلة حول ال multiplicité :

- نسخة واحدة فقط:1 أو 1..1

- صفر نسخة أو أكثير: * أو *..0

- على الأقل نسخة واحدة: *..1

- من واحد إلى ستة نسخ : 1..6

- اتجاه الرابطة La navigabilité

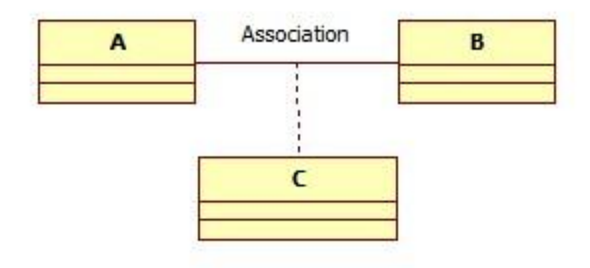
في الحالة العامة اتجاه الرابطة يكون في الجهتين ، لكن يمكننا تحديد الاتجاه في حالة كون فئة ما تستعمل فئة أخرى و الفئة الثانية لا يمكنها استعمال الفئة الأولى ، و هذا ما نلاحظه في المثال السابق حيث أن ال Polygone معرف بثلاث نقاط بينما النقطة من المستحيل أن تكون معرفة ب Polygone و هذا ما يدل عليه اتجاه السهم في نهاية الرابطة .

يجدر بي الإشارة إلى أن الرابطة لا تكون فقط بين فئتين اثنين بل يمكنها أن تتعدى ذلك إلى ثلاث فئات أو أكثر و تسمى في هذه الحالة رابطة متعددة Association n-aire، بينما تسمى في الحالة الأولى Association binaire رابطة ثنائية.

- الفئة-الرابطة La classe-association :

الفئة-الرابطة تجمع بين خصائص الفئات و خصائص الروابط معا، فهي تقوم بربط فئتين أو أكثر و تحتوي صفات و عمليات Des attributs et des opérations مثل أي فئة أخرى .

تمثل الفئة-الرابطة بخط متقطع بين الفئة و الرابطة التي تمثلها و هذا شكلها العام:

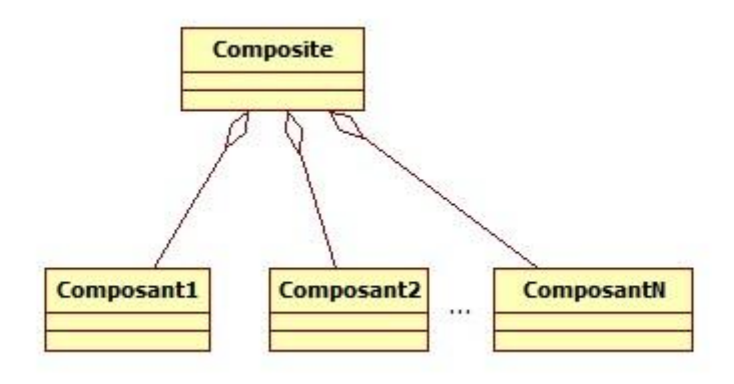


حيث أن الفئة C هي الفئة-الرابطة.

- التجميع L'agrégation:

التجميع هي حالة خاصة من الروابط تعبر عن علاقة احتواء ، هذه العلاقة لا تحتاج لتسمية فهي تعني <<يحتوي>> أو <<مشكل من>> . تمثل هذه العلاقة بمعين فارغ يشير إلى الفئة الحاوية.

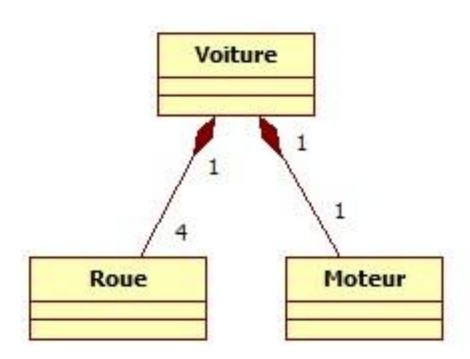
هذا مخطط عام هذه العلاقة :



- التركيب La composition :

التركيب أقوى من التجميع إذ أن الكائن المشكل من تركيب عدة كائنات أخرى لا يمكنه الاستغناء عن كائن واحد من هذه الكائنات التي تشكله. تمثل هذه العلاقة بمعين ممتلئ يشير إلى الفئة المركَبَة.

مثال ذلك السيارة مركبة من محرك moteur و أربع عجلات quatre roues ، فإذا نزعنا مركب من هذه المركبات من السيارة لا يمكننا القول عنها أنها سيارة كاملة .



3.2 إعداد مخطط الفئات:

من أجل بناء مخطط الفئات علينا إتباع الخطوات التالية :

جد فئات المجال الذي أنت بصدد دراسته: هذه المرحلة تتم غالبا باستشارة خبير في هذا المجال فهو يزودك بأدق التفاصيل و يعطيك كل المعلومات التي تحتاجها من أجل استخراج الفئات و خصائصها ، و يتم ذلك غالبا عن طريق استعمال دفتر الشروط Cahier des charges تكتب فيه جميع معلومات المشروع التي نستقيها من المستخدم .و في حالتنا نحن علينا الاتصال بصاحب المحل الذي يزودنا بكل المعلومات اللازمة لنكتبها هنا في المنتدى في فقرة نص المشروع ، بعدها علينا التمحيص في هذه المعلومات و استخلاص الفئات التي نحتاجها .

جد العلاقات أو الروابط بين الفئات: العلاقات دوما تستخرج من أفعال أو من جمل فعلية تدل على ارتباط فئة بفئة أخرى أو بعدد من الفئات ، مثل <<est composé de>> (مركب من) ، <<Possède>> (يملك) ، <<Travail pour >> (يعمل لدى) ...

جد خصائص الفئات: الصفات Attributs غالبا ما تكون عبارة عن أسماء أو جمل سمية كسرعة السيارة مثلا

<<la vitesse d'une voiture>>، أو سعر منتج <<Prix d'un produit>> .

بالنسبة للعمليات فيمكننا جمعها من خلال معرفة مختلف الوظائف التي يؤديها كائن ما ، كإضافة منتج جديد Ajouter produit ، تعديل منتج ...

نظم و بسط النموذج : و ذلك عن طريق إزالة الفئات المكررة و استخدام الوراثة.

لا يمكننا بناء مخطط فئات كامل من أول وهلة بل سنضطر في كل مرة تظهر لنا فيها مستجدات للعودة إلى هذا المخطط أو أي مخطط آخر للتعديل عليه و إضافة كل ما هو جديد. و عليه فإن النمذجة كائنية المنحى هي عبارة عن عملية تكرارية و ليست خطية ، سنضطر فيها للعودة للوراء حال اضطرارنا لذلك.

4.2 تحويل مخطط الفئات إلى تعليمات برمجية :

لكل من التصاميم و الرسومات السابقة ترجمة في لغات البرمجة ، فهدفنا في النهاية الوصول إلى البرنامج الذي نحن بصدد دراسته و هذا لا يكون إلا باستعمال لغة من لغات البرمجة ، طبعا كائنية المنحى . في القسم الموالي سأقوم بذكر كيفية تحويل مخطط الفئات في لغة البرمجة كائنية المنحى سي شارب (C#) :

تحويل فئة:

```
class ClasseA
    {
        public string Attribut1;
        private int Attribut2;
        protected double Attribut3;
        ...

        public void opération1;
        private string opération2;
        ...
    }
```

ClasseA
+Attribut1: string
-Attribut2: int
#Attribut3: double
+...
+Operation1()
-Operation2(): string
+...()

تحويل علاقة وراثة:

```
class ClasseA:ClasseB
    {
        ...
        ...
    }
                 }
```

ClasseA
ClasseB

تحويل رابطة 1 مقابل 1:

```
class ClasseA
    {
        public ClasseB b = new ClasseB();

    }
  class ClasseB
    {
          public ClasseA a = new ClasseA();

    }
```

ClasseA +a 1 — 1 +b ClasseB

تحويل رابطة 1 مقابل 1 ذات اتجاه واحد:

```
class ClasseA
    {
        private ClasseB b = new ClasseB();

    }
 class ClasseB
    {
          // هذه الفئة لا تحتوي على نسخ من الفئة A
          ...

    }
```

ClasseA 1 → 1 -b ClasseB

تحويل رابطة 1 مقابل *:

```
class ClasseA
    {
       private ArrayList<ClasseB> b = new ArrayList();

    }
      class ClasseB
    {
         public ClasseA a = new ClasseA();

    }
```

تحويل رابطة 1 مقابل * ذ ات اتجاه واحد:

```
class ClasseA
    {
       private ArrayList<ClasseB> b = new ArrayList();

    }


class ClasseB
    {
         // A الفئة من نسخ تحتوي لا الفئة هذه

    }
```

تحويل رابطة 1 مقابل قيمة محددة :

```
class ClasseA
    {
        private ClasseB[] b = new ClasseB[8];

    }

class ClasseB
    {
          public ClasseA a = new ClasseA();
    }
```

تحويل رابطة التجميع :

يكون تحويل هذه الرابطة كتحويل أي رابطة عادية

تحويل رابطة التركيب :

تحول رابطة التركيب يكون كأي رابطة عادية ذات اتجاه واحد



بعد كل هذا الدرس الطويل و الممل سنعود الآن لمشروعنا لنطبق كل ما درسناه عليه ، لكن قبل ذلك سنضطر في هذه المرة أيضا للعودة إلى نص مشروعنا للتعديل عليه قليلا كالعادة . التعديلات و الإضافات موضحة باللون الأخضر.

1 . المحل يحتوي على منتجات للبيع Produits تشكل لنا المخزون المتوفر في المحل.
2. يقوم صاحب المحل بشراء منتجات من عند ممولين fournisseurs و ذلك في تاريخ محدد و عن طريق فاتورة تقدم من طرف الممول و يتم إدخال بيانات هذه المنتجات كوصول Arrivage.
3. قبل شراء المنتجات صاحب المحل يقوم بتحرير طلبية Commande يضع فيها جميع المنتجات التي يريد شرائها من عند الممول.
4. يتعامل صاحب المحل مع الزبائن عن طريق الفواتير التي تحوي المنتجات المشترات .
5. كل منتج لديه تعريفة أو رمز Code لديه وصف Désignation, ثمن شراء Prix Achat, ثمن بيع Prix Vente, كمية Quantité و الحد الأدنى في المخزن Stock Minimum.
6. الفاتورة تحتوي على العديد من المنتجات، في المقابل المنتج الواحد يمكن أن يظهر في أكثر من فاتورة. الفاتورة لديها رقم Numéro , تاريخ إجراء الفاتورة Date de facture و قيمة المبلغ المدفوع للفاتورة Montant
7. الزبون يتصف برقم، اسم، لقب، رقم الهاتف، عنوان .
8.الطلبية تحتوي رقم و تاريخ إجراء الطلبية.

9. الممول لديه رقم، اسم، لقب، رقم هاتف، فاكس، عنوان.
10. طبعا كل عملية تحديث أو إضافة أو حذف لمنتج أو فاتورة أو زبون أو ممول يقوم بها صاحب المحل بعدما يسجل دخوله للبرنامج بكلمة سر .
11. يمكن لصاحب المحل من تعديل إعدادات البرنامج من تغيير للغة البرنامج أو أخذ نسخة احتياطية من قاعدة البيانات أو الرجوع إلى نسخة تم تخزينها من قبل.
12. يمكن لصاحب المحل الاضطلاع على المخزن و ذلك من أجل تحديد المنتجات التي على وشك النفاذ منه أو من أجل تعديل بيانات محددة.

13. كل منتجا ينتمي لفئة محددة من المنتجات.

14. دفع الفاتورة يكون دفعة واحدة أو على مراحل و يتم في كل مرحلة تسجيل تاريخ الدفع و المبلغ المدفوع. نسمي هذه العملية بالـ versement . كذلك بالنسبة للمنتجات التي يجلبها صاحب المحل من عند الممولين يمكنها أن تكون دفعة واحدة أو على مراحل.

15. يمكن أن يكون هناك أكثر من مدير للمحل يتحكم في مختلف عمليات البيع و الشراء أو غيرها .

16. يمكن لصاحب المحل أن يقوم بصرف بعض التكاليف,Dépense نقوم بتخزين تاريخ صرفها و قيمة هذه التكاليفLe montant.

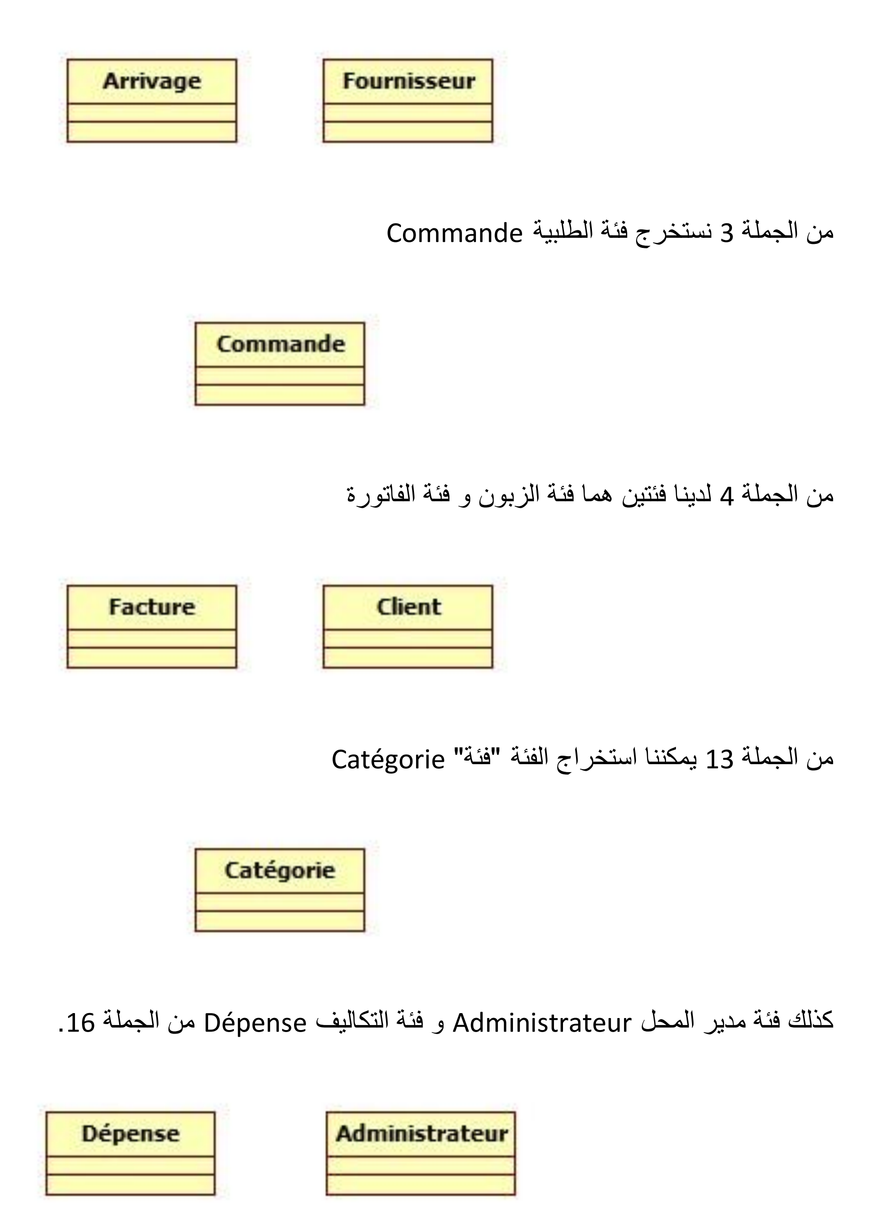
الآن نبدأ في نمذجة مخطط الفئات اعتمادا على المراحل التي ذكرناها في الدرس :

<u>أولا استخراج الفئات:</u>

من الجملة 1 من نص المشروع نستخرج الفئة منتج Produit

Produit

من الجملة 2 يمكننا استخراج فئة الممول Fournisseur و فئة الـ Arrivage



من الجملة 3 نستخرج فئة الطلبية Commande

من الجملة 4 لدينا فئتين هما فئة الزبون و فئة الفاتورة

من الجملة 13 يمكننا استخراج الفئة "فئة" Catégorie

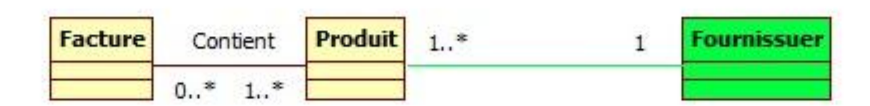
كذلك فئة مدير المحل Administrateur و فئة التكاليف Dépense من الجملة 16.

ثانيا استخراج الروابط بين الفئات :

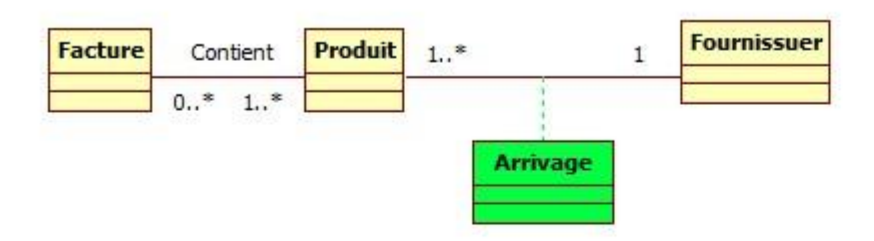
أولا الفئة Produit هي الفئة المركزية في شروعنا لأن أغلب الروابط تكون بينها و بين الفئات الأخرى و بالرجوع إلى الجملة 4 من نص المشروع نستخرج رابطة بين الفئة Produit و الفئة Facture فكل فاتورة تحتوي منتج واحد على الأقل أو أكثر، و كل منتج يمكن أن يظهر في أي فاتورة. هذا ما يمكننا ترجمته في المخطط التالي :

Facture — Contient — Produit
0..* 1..*

من الجملة 2 نستخلص رابطة بين المنتج و الممول حيث أن لكل منتج في مخزننا ممول أحضرنا من عنده هذا المنتج ، حيث لكل منتج ممول واحد و لكل ممول العديد من المنتجات التي قام ببيعها لصاحب المحل .



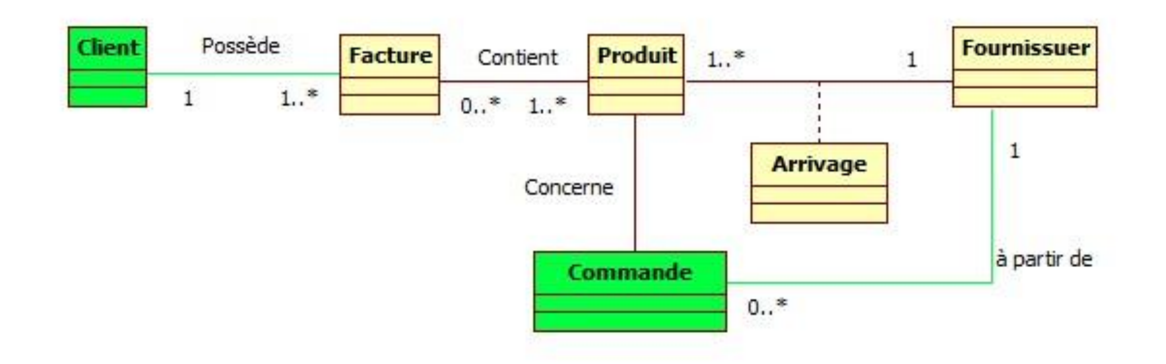
الرابطة التي بين الممول و المنتج هي رابطة وصول Arrivage، هذه الرابطة تحتوي على خصائص كما ذكرنا في نهاية الجملة الثانية لذلك علينا تحويلها إلى فئة_رابطة ليصبح المخطط هكذا:



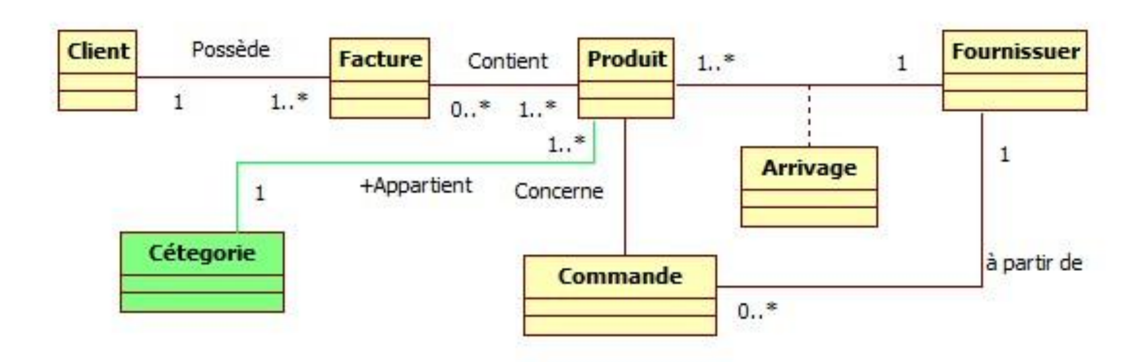
من الجملة 3 يمكننا استخراج رابطة احتواء بين الطلبية و المنتج ، الطلبية و الممول ، حيث يقوم صاحب المحل بتحرير طلبية يضع فيها جميع المنتجات التي يعتزم شرائها من عند ممول يحدده مسبقا .

من الجملة 5 يمكننا تحديد رابطة بين الزبون و الفاتورة ، حيث لكل زبون فاتورة أو أكثر و الفاتورة متعلق بزبون واحد و وحيد.

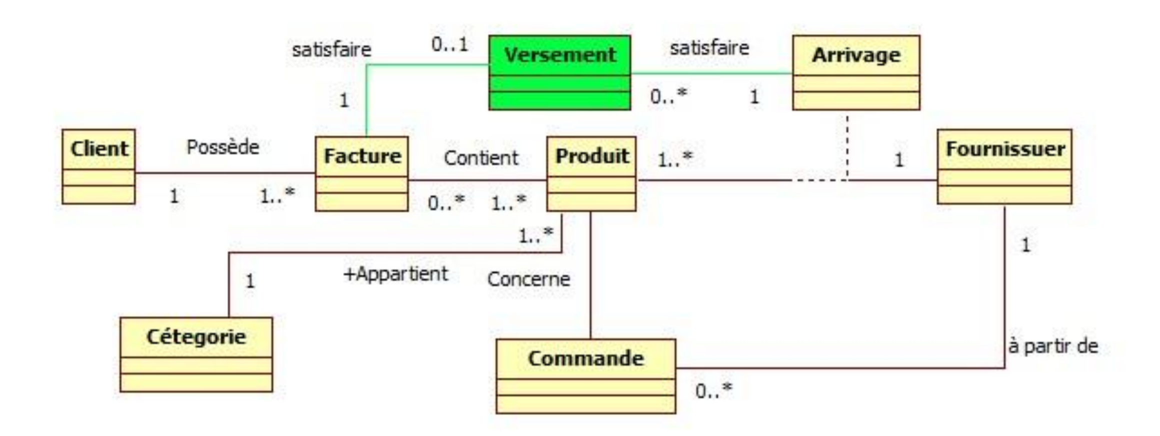
فيكون مخططنا كما يلي :



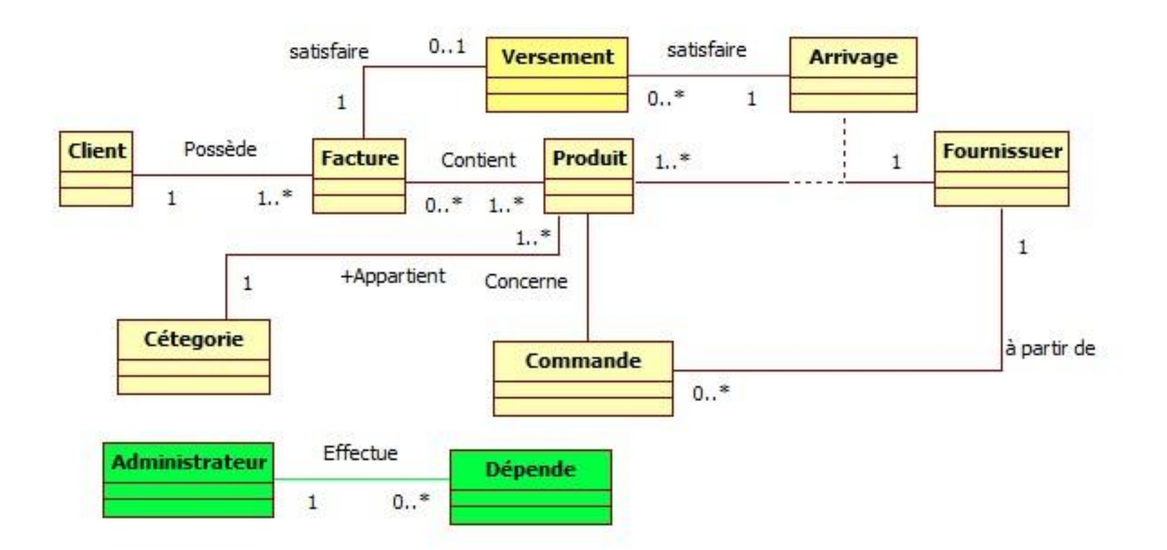
من الجملة 13 كل منتج ينتمي لفئة معينة Catégorie ، أي توجد رابطة انتساب Appartenace بين فئة Produit و فئة Catégirie :



الجملة 14 : قبل وصولي إلى هذا الحل أي إنشاء فئة Versement بها كل هذه الخصائص. كانت هذه الفئة عبارة عن صفة من صفات فئة ال Facture و فئة Arrivage ، و كانت مهيئة للاحتواء قيمة الفاتورة فقط ، لكن بعض التواصل مع صاحب المحل اتضح أن الـ Versement يمكن أن يكون على مراحل و يمكن أن تحمل كل مرحلة من الدفع قيمة معينة . لذا ستكون هذه الفئة مرتبطة مع الفئة Facture و الفئة Arrivage على حدا سواء:



الجملة 16 : ستكون علاقة الـ administrateur جد واضحة مع المصاريف Dépenses ،حيث يمكن أن يقوم أي Administrateur بصرف تكاليف معينة.



ثالثا استخراج خصائص الفئات :

اعتمادا على النص أعلاه و بالرجوع إلى صاحب المحل ، الجدول التالي يحتوي كل الفئات و خصائصها :

opération العملية	Attribut الصفة	Classe الفئة
AjouterProd() ModifierProd() SupprimerProd() AfficherProd()	Code_Prod Designation PrixAchat PrixUnitaire QuantitéEnStock StockMin TVA	Produit
AjouterFacture() SupprimerFacture() ModifierFacture() AnnulerFacture() AfficherFacture() ImprimerFacture()	Num_Fact Date	Facture
AjouterFourn() SupprimerFourn() ModifierFourn()	ID_Fourn Nom Prenom Tel Fax Adresse Banque Num_compte	Fournisseur
AjouterArriv() ModifierArriv() SupprimerArriv() AnnulerArriv()	Num_Arriv Date Quantité PrixAchat	Arrivage

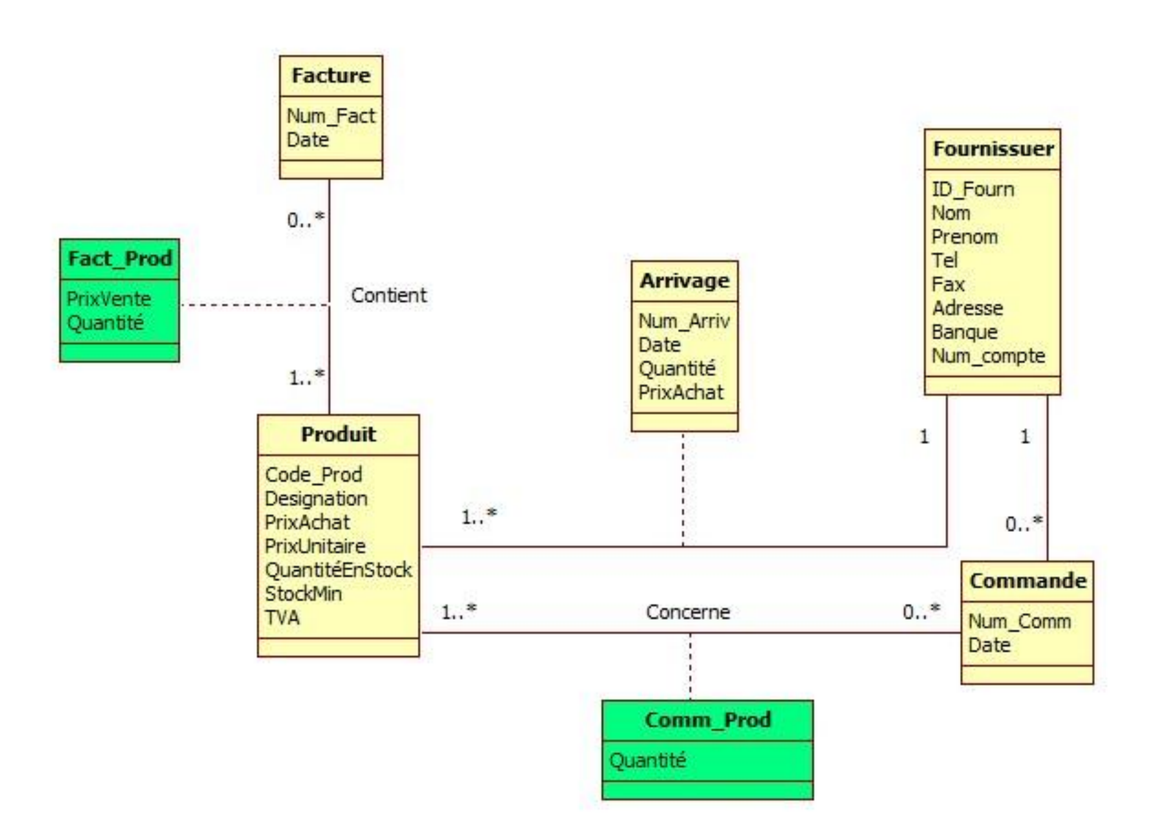
AjouterComm() **ModifierComm()** **SupprimerComm()** **ImprimerComm()**	**Num_Comm** **Date**	**Commande**
AjouterClient() **SupprimerClient()** **ModifierClient()** **AfficherClients()** **AfficherCréditClient()**	**ID_Cient** **Nom** **Prenom** **Tel** **Adresse** **Banque** **Num_Chèque**	**Client**
AjouterCat **SupprimerCat** **modifierCat**	**Num_Cat** **Nom_Cat** **Desc_cat**	**Catégorie**
AjouterVersement() **SupprimerVersement()** **ModifierVersement()**	**Num_Versement** **Date_Versement** **Montant**	**Versement**
AjouterAdmin() **ModifierAdmin()** **SupprimerAdmin()**	**Nom_Admin** **MotDePasse**	**Administrateur**
AjouterDép() **ModifierDép()** **ModifierDép()** **SupprimerDép()**	**Num_Dép** **Date** **Montant**	**Dépense**
	PrixVente **Quantité**	**Fact_Prod**
	Quantité	**Comm_Prod**

هناك توضيح فيما يخص الفئة Arrivage ، هذه الفئة تحتوي على صفتين ربما يتضح أنها ليستا تابعتين لها بل خاصتين بالفئة Produit .

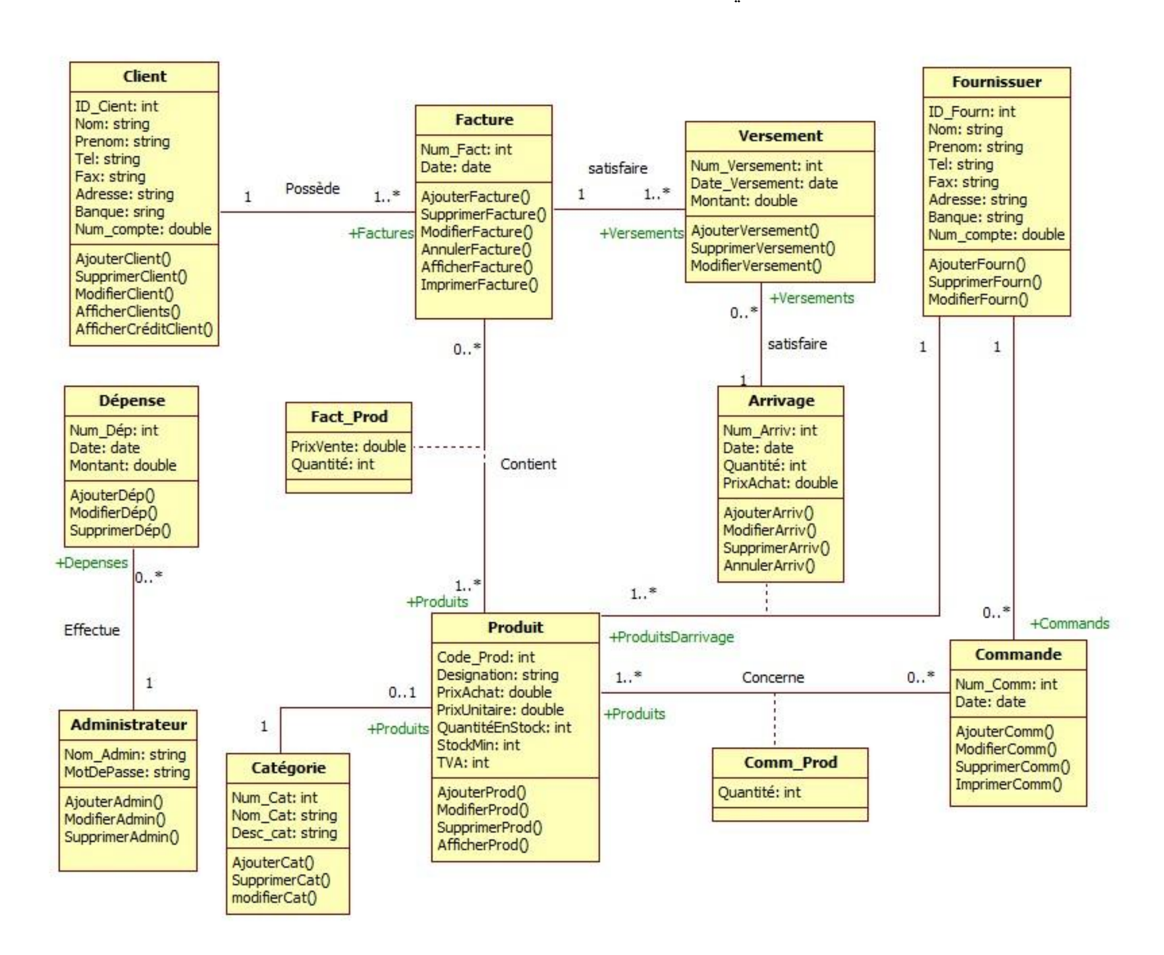
لنفهم سبب وضعهما هنا علينا أولا الاتفاق على أمر مهم ذكرناه في نص المشروع ، لقد قلنا أن المنتجات تمثل لنا المخزون المتوفر في المحل ، أي أن الفئة Produit تمثل بشكل من الأشكال المخزن في حد ذاته . حيث تحدد الصفة Quantité كمية هذا المنتج المتوفرة في المحل بينما الصفة Quantité الموجودة في الفئة Arrivage فهي خاصة بهذه الفئة و تخص منتجا معينا من منتجات المخزن ، كذلك الأمر بالنسبة للصفة PrixAchat، هذه الصفة تمثل سعر **الشراء** العام ، أي الذي حسبناه اعتمادا على مجموعة من الإيصالات Arrivages ، هذا في الفئة Produit أما في الفئة Arrivage فهذه الخاصية تمثل سعر شراء المنتج من عند الممول .

أيضا قد أضفت فئتين جديدتين هما Fact_Prod ، Comm_Prod لسبب مماثل لما سبق و هما فئتين تربطان المنتج و الفاتورة ، الطلبية و المنتج على التوالي. حيث من المعلوم أنه إذا جاء مشتري لشراء منتج ما فسعر بيع هذا المنتج لهذا الزبون ليس بالضرورة يكون مطابقا للسعر الأولي الموضوع لهذا المنتج في المخزن ، كذلك الأمر بالنسبة للخاصيتين Quantité الموجودتين في الفئتين Facture و Commande فهذه الكمية خاصة بالفاتورة أو خاصة بالطلبية و هي مختلفة من فاتورة إلى فاتورة كما أنها مغايرة للكمية الموجودة في المخزن . و يمكنك استنتاج نفس الشيء بالنسبة للخاصية PrixVente.

للتوضيح هذا جزء من المخطط بعد إضافة الفئتين الجديدتين:

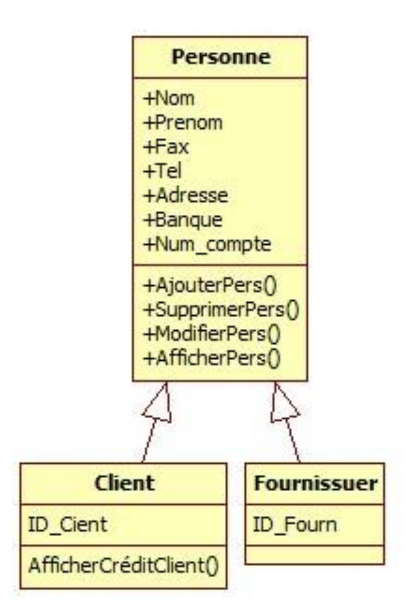


بعد إضافة كل هذه الخصائص للفئات وضع أنواع هه الخصائص(String ، int ...) و إضافة الفئتين و أطراف الروابط الملونة بالأخضر نحصل على المخطط التالي :



المرحلة الأخيرة : تبسيط و تنظيم المخطط :

يمكننا تحسين المخطط عن طريق استخراج الفئات التي يمكننا أن نربط بينها بعلاقة الوراثة. لو تأملت معي قليلا لوجدت أن هذه العلاقة يمكن عملها بين فئات يمكنها أن تحمل خصائص مشتركة فيما بينها. هذه الفئات هي فئة Client و Fournisseur حيث سنجمع الخصائص المشتركة بين هذه الفئات في فئة جديدة اسمها Personne و سنُوَرِّث هذه الفئة من الفئتين السابقتين . المخطط التالي يوضح علاقة الوراثة بين هذه الفئات الثلاثة :



ليكون مخططنا في النهاية كما يلي:

Personne
Nom: string
Prenom: string
Tel: string
Fax: string
Adresse: string
Banque: string
Num_compte: double
AjouterPers()
SupprimerPers()
ModifierPers()
AfficherPers()

Client
ID_Cient: int
AfficherCréditClient()

1 Possède 1..*
+Factures

Facture
Num_Fact: int
Date: date
AjouterFacture()
SupprimerFacture()
ModifierFacture()
AnnulerFacture()
AfficherFacture()
ImprimerFacture()

satisfaire
1 1..*
+Versements

Versement
Num_Versement: int
Date_Versement: date
Montant: double
AjouterVersement()
SupprimerVersement()
ModifierVersement()

Fournissuer
ID_Fourn: int

1
1

+Versements
0..*
satisfaire
1

Arrivage
Num_Arriv: int
Date: date
Quantité: int
PrixAchat: double
AjouterArriv()
ModifierArriv()
SupprimerArriv()
AnnulerArriv()

0..*

Dépense
Num_Dép: int
Date: date
Montant: double
AjouterDép()
ModifierDép()
SupprimerDép()

Fact_Prod
PrixVente: double
Quantité: int

Contient

+Depenses
0..*
Effectue
1

1..*
+Produits
1..*
0..*
+Commands
+ProduitsDarrivage

Produit
Code_Prod: int
Designation: string
PrixAchat: double
PrixUnitaire: double
QuantitéEnStock: int
StockMin: int
TVA: int
AjouterProd()
ModifierProd()
SupprimerProd()
AfficherProd()

1..* Concerne 0..*
+Produits

Commande
Num_Comm: int
Date: date
AjouterComm()
ModifierComm()
SupprimerComm()
ImprimerComm()

Administrateur
Nom_Admin: string
MotDePasse: string
AjouterAdmin()
ModifierAdmin()
SupprimerAdmin()

1
0..1
+Produits

Catégorie
Num_Cat: int
Nom_Cat: string
Desc_cat: string
AjouterCat()
SupprimerCat()
modifierCat()

Comm_Prod
Quantité: int

طبعا لا يمكننا ضبط جميع هذه الخصائص من أول نمذجة كما قلت سابقا فربما ينقصنا صفات أو عمليات أخرى أو هناك صفات أو عملية لا حاجة لنا بها ، لذا سنضطر للتعديل على هذا الجدول كلما احتجنا لذلك .

كذلك الأمر بالنسبة للفئات سنقوم بتحديثها كلما دعت الضرورة لذلك.

إلى هنا أقوم قد أنهيت هذا الجزء من تصميم المشروع و أرجو أن لا أكون قد أطلت عليكم.

أي اقتراحات أو استفسارات أو مداخلات نحن هنا لمناقشتها جميعا.

بالنسبة لهذا المخطط فقد استعملت برنامج StarUML و هو أفضل من البرنامج السابق بكثير. لتحميل البرنامج اضغط هذا الرابط:

http://www.mediafire.com/?uqmyyynlb2z

أو زر هذا الرابط للاضطلاع على موضوع البرنامج:

http://www.science-collector.com/vb/t3832.html

المخططات موجودة في المرفقات لمن يريد تحميلها و هي تحتاج للبرنامج المذكور أعلاه لفتحها.

كذلك أضفت نسخة PDF لتحميل الموضوع و قراءته على شكل كتاب إلكتروني.